

Program Dinamis

Bahasan

- **Definisi algoritma program dinamis**
- **Contoh-contoh kasus Program Dinamis**
 1. **Lintasan Terpendek**
 2. **MCM**
 3. **LCS**
 4. **Penganggaran Modal**
 5. **TSP**

Program Dinamis

❖ Program Dinamis (dynamic programming):

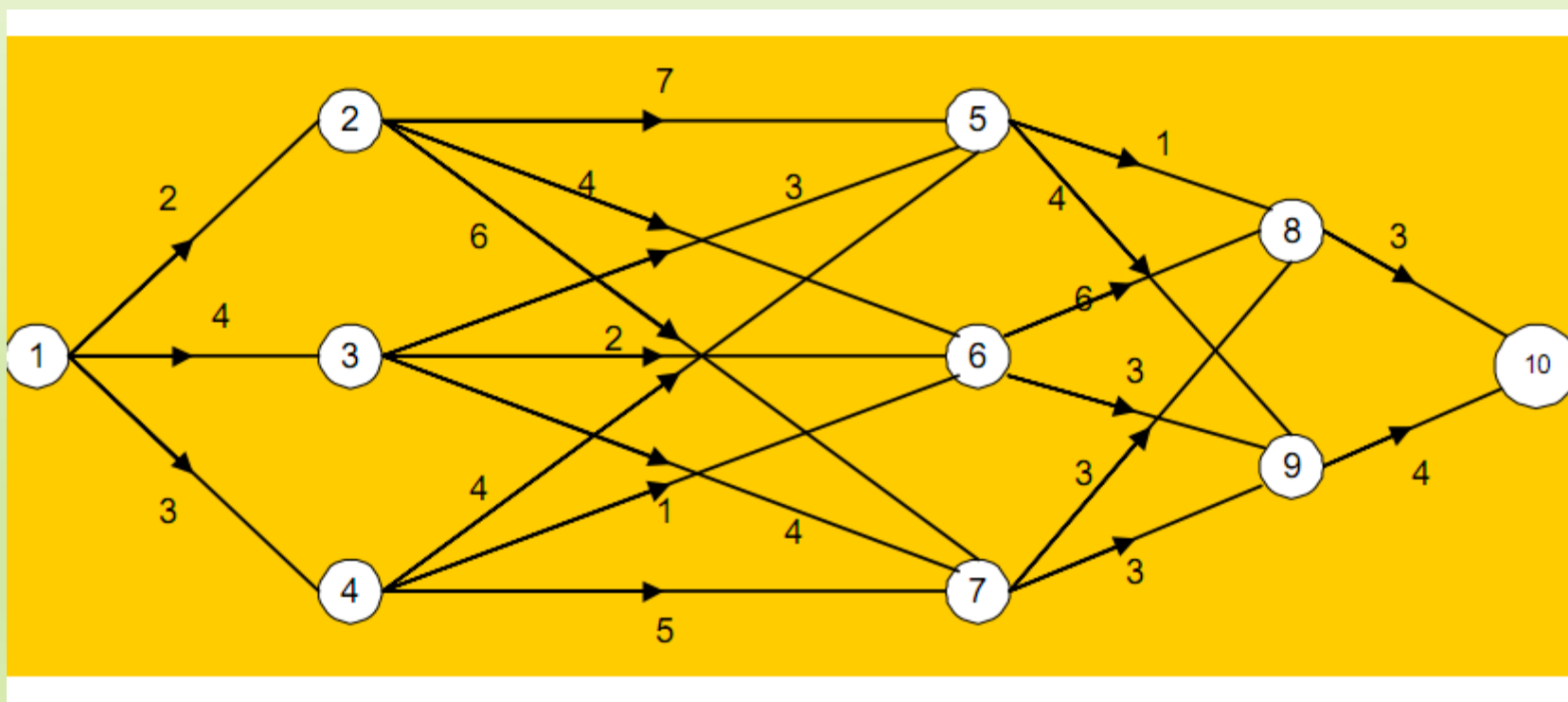
- ❖ metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (step) atau tahapan (stage).
- ❖ sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan.

❖ Penyelesaian persoalan dengan metode ini:

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Program Dinamis

Tinjau graf di bawah ini. Kita ingin menemukan lintasan terpendek dari 1 ke 10



Prinsip Optimalitas

- ❖ Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas.
- ❖ Prinsip Optimalitas: jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal.
- ❖ Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap k ke tahap k + 1, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal.
- ❖ $\text{Ongkos pada tahap } k + 1 = (\text{ongkos yang dihasilkan pada tahap } k) + (\text{ongkos dari tahap } k \text{ ke tahap } k + 1)$

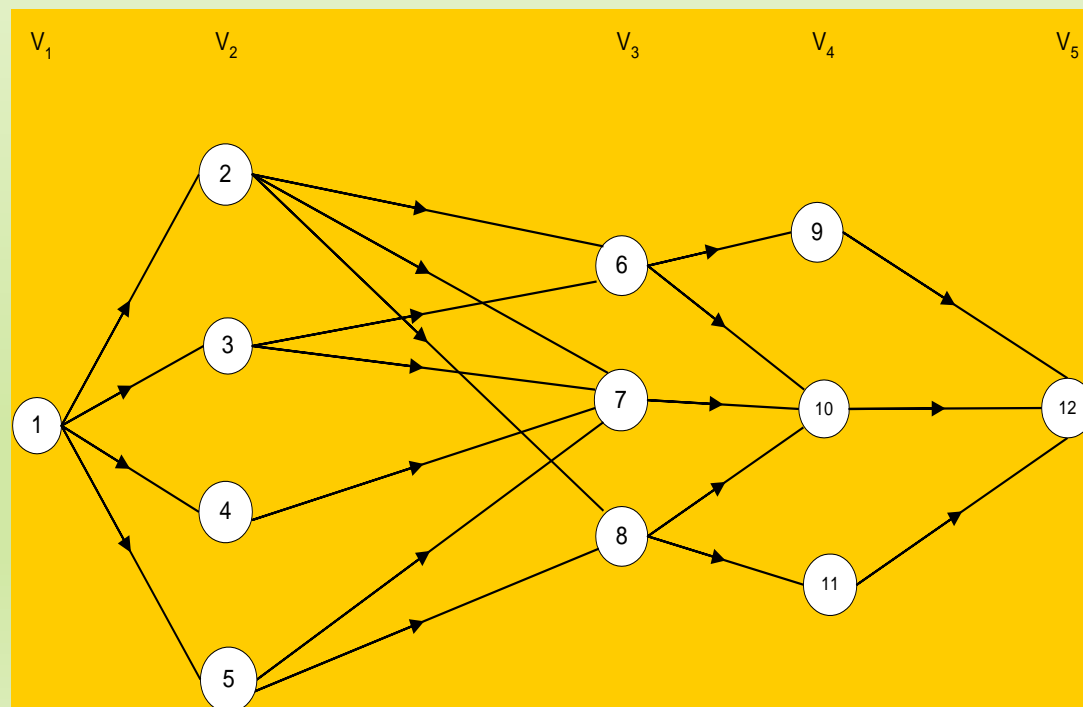
Karakteristik Persoalan Program Dinamis

1. Persoalan dapat dibagi menjadi beberapa tahap (stage), yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (state) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.

Karakteristik Persoalan Program Dinamis

6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasi keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap $k + 1$.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Graf multistage (multistage graph). Tiap simpul di dalam graf tersebut menyatakan status, sedangkan V_1, V_2, \dots menyatakan tahap.



Dua Pendekatan Program Dinamis

- ❖ Dua pendekatan yang digunakan dalam PD: maju (*forward* atau *up-down*) dan mundur (*backward* atau *bottom-up*).
- ❖ Misalkan x_1, x_2, \dots, x_n menyatakan peubah (*variable*) keputusan yang harus dibuat masing-masing untuk tahap 1, 2, ..., n . Maka,
 - ❖ Program dinamis maju. Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap n . Runtunan peubah keputusan adalah x_1, x_2, \dots, x_n .
 - ❖ Program dinamis mundur. Program dinamis bergerak mulai dari tahap n , terus mundur ke tahap $n - 1, n - 2$, dan seterusnya sampai tahap 1. Runtunan peubah keputusan adalah x_n, x_{n-1}, \dots, x_1 .

Pengembangan Algoritma Program Dinamis

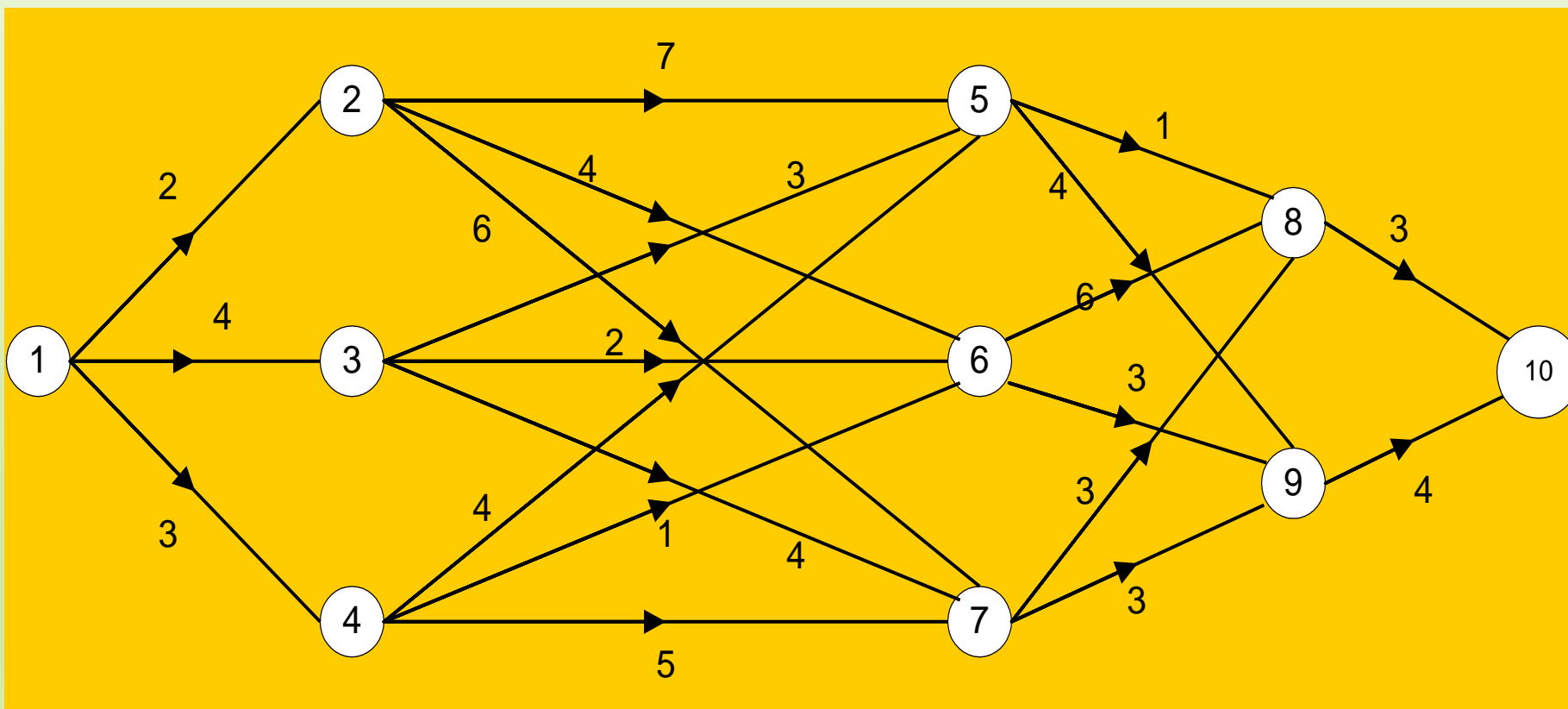
❖ Langkah-langkah Pengembangan Algoritma Program Dinamis

1. Karakteristikkan struktur solusi optimal.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur.
4. Konstruksi solusi optimal.

Contoh-contoh kasus PD

1. Lintasan Terpendek (Shortest Path)

Tentukan lintasan terpendek dari simpul 1 ke simpul 10:



1. Lintasan Terpendek dengan PD Mundur

- ❖ Misalkan x_1, x_2, \dots, x_4 adalah simpul-simpul yang dikunjungi pada tahap k ($k = 1, 2, 3, 4$).
- ❖ Maka rute yang dilalui adalah $1 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$, yang dalam hal ini $x_4 = 10$.

Pada persoalan ini,

- ❖ *Tahap* (k) adalah proses memilih simpul tujuan berikutnya (ada 4 tahap).
- ❖ *Status* (s) yang berhubungan dengan masing-masing tahap adalah simpul-simpul di dalam graf.

1. Lintasan Terpendek dengan PD Mundur

Relasi rekurens berikut menyatakan lintasan terpendek dari status s ke x_4 pada tahap k :

$$f_4(s) = c_{sx_4} \quad (\text{basis})$$

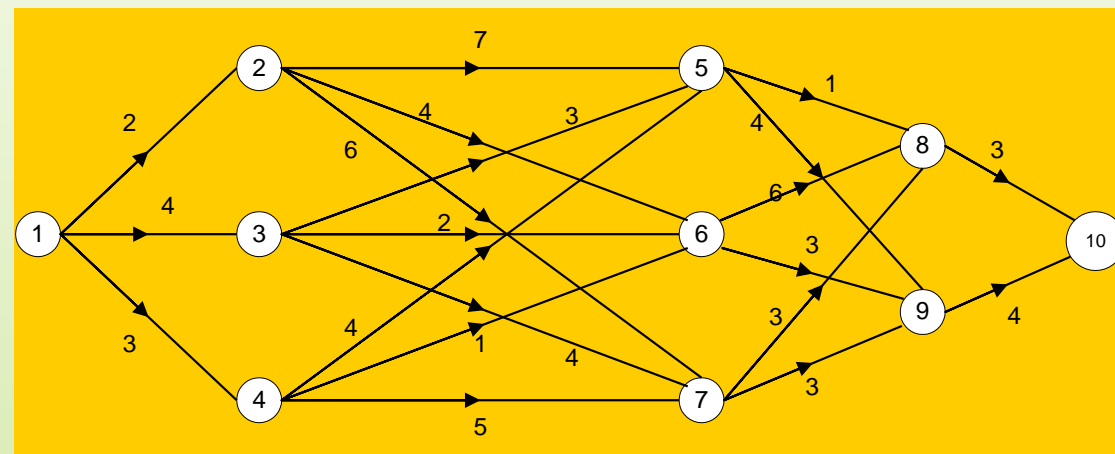
$$f_k(s) = \min_{x_k} \{c_{sx_k} + f_{k+1}(x_k)\}, \quad (\text{rekurens})$$

$k = 1, 2, 3$

Keterangan:

- x_k : peubah keputusan pada tahap k ($k = 1, 2, 3$).
- c_{sx_k} : bobot (*cost*) sisi dari s ke x_k
- $f_k(s, x_k)$: total bobot lintasan dari s ke x_k
- $f_k(s)$: nilai minimum dari $f_k(s, x_k)$

Tujuan program dinamis mundur: mendapatkan $f_1(1)$ dengan cara mencari $f_4(s), f_3(s), f_2(s)$ terlebih dahulu.



Tahap 4:

$$f_4(s) = c_{sx_4}$$

s	Solusi Optimum	
	$f_4(s)$	x_4^*
8	3	10
9	4	10

Catatan: x_k^* adalah nilai x_k yang meminimumkan $f_k(s, x_k)$.

1. Lintasan Terpendek dengan PD Mundur

Tahap 4:

$$f_4(s) = c_{sx_4}$$

s	Solusi Optimum	
	$f_4(s)$	x_4^*
8	3	10
9	4	10

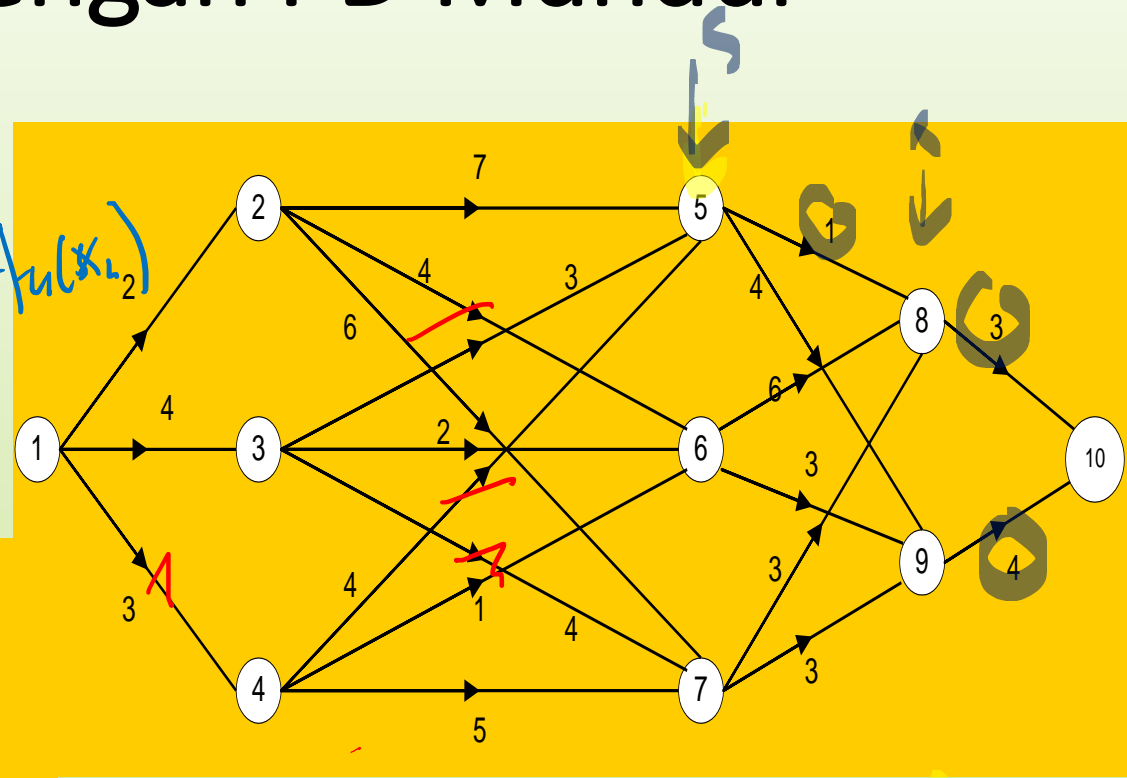
$f_3(s, x_4) = \min \{ c_{sx_3} + f_4(x_3) \}$

Catatan: x_k^* adalah nilai x_k yang meminimumkan $f_k(s, x_k)$.

Tahap 3:

$$f_3(s) = \min_{x_3} \{ c_{sx_3} + f_4(x_3) \}$$

x_3 \ s	$f_3(s, x_3) = c_{sx_3} + f_4(x_3)$		Solusi Optimum	
	8	9	$f_3(s)$	x_3^*
5	4	8	4	8
6	9	7	7	9
7	6	7	6	8



$f_2(s) = \min \{ c_{sx_2} + f_3(x_2) \}$

$f_1(s) = \min \{ c_{sx_1} + f_2(x_1) \}$

1. Lintasan Terpendek dengan PD Mundur

Tahap 3:

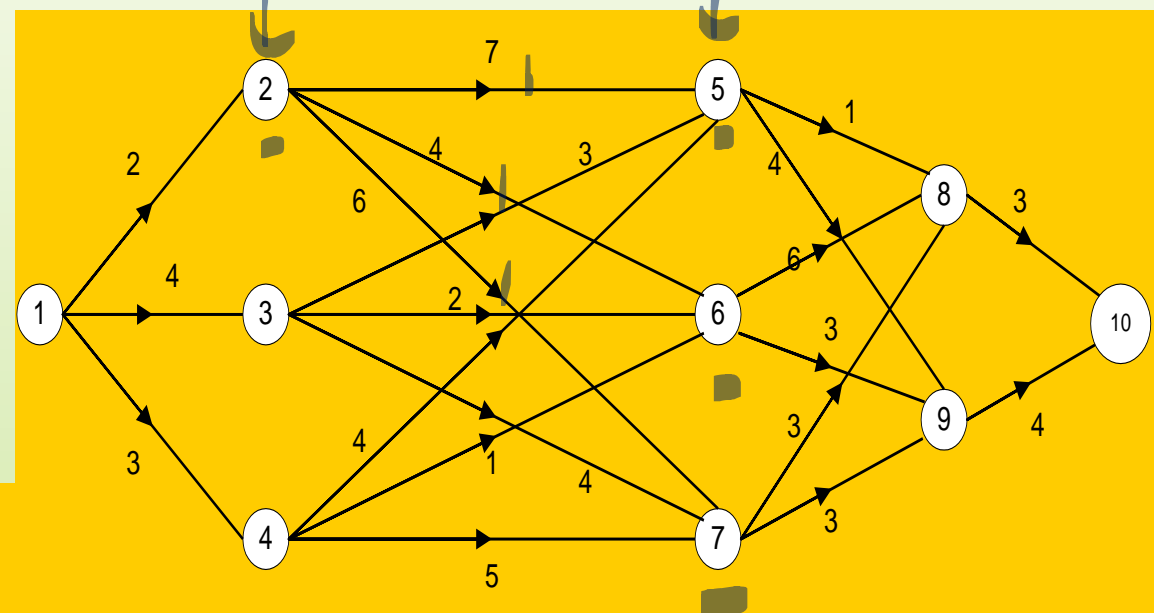
$$f_3(s) = \min_{x_3} \{c_{sx_3} + f_4(x_3)\}$$

x_3	$f_3(s, x_3) = c_{sx_3} + f_4(x_3)$		Solusi Optimum	
$s \backslash$	8	9	$f_3(s)$	x_3^*
5	4	8	4	8
6	9	7	7	9
7	6	7	6	8

Tahap 2:

$$f_2(s) = \min_{x_2} \{c_{sx_2} + f_3(x_2)\}$$

x_2	$f_2(s, x_2) = c_{sx_2} + f_3(x_2)$			Solusi Optimum	
$s \backslash$	5	6	7	$f_2(s)$	x_2^*
2	11	11	12	11	5 atau 6
3	7	9	10	7	5
4	8	8	11	8	5 atau 6



1. Lintasan Terpendek dengan PD Mundur

Tahap 2:

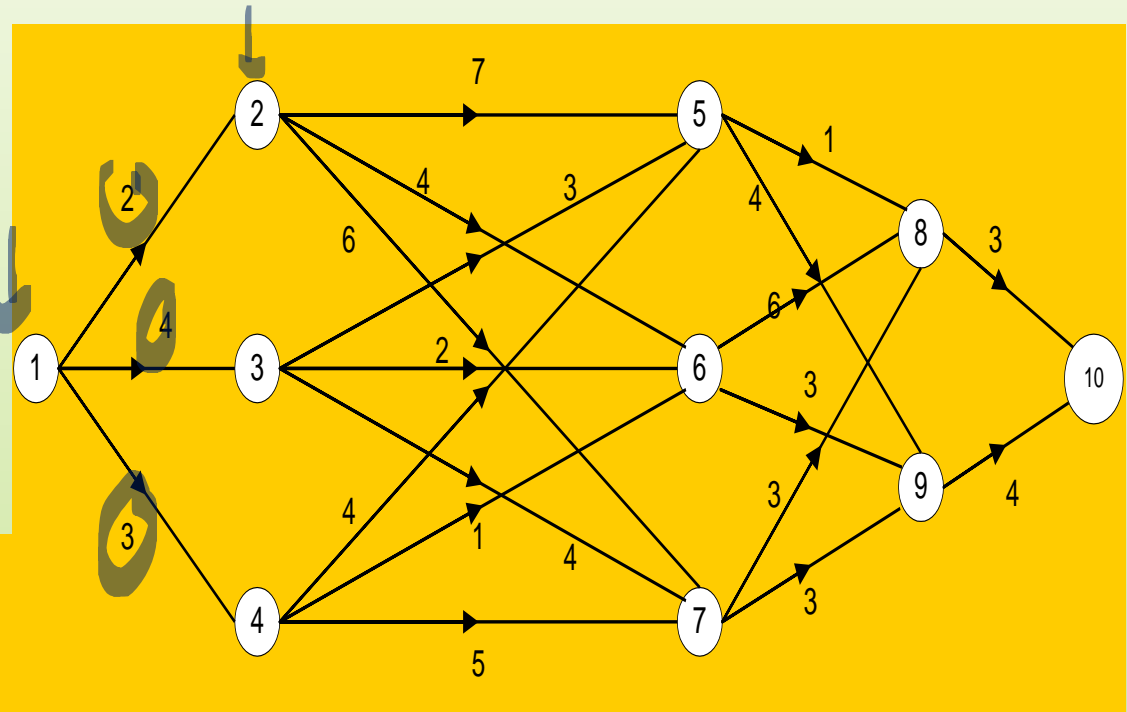
$$f_2(s) = \min_{x_2} \{c_{sx_2} + f_3(x_2)\}$$

x_2	$f_2(s, x_2) = c_{s,x_2} + f_3(x_2)$			Solusi Optimum	
$s \backslash$	5	6	7	$f_2(s)$	x_2^*
2	11	11	12	11	5 atau 6
3	7	9	10	7	5
4	8	8	11	8	5 atau 6

Tahap 1:

$$f_1(s) = \min_{x_1} \{c_{sx_1} + f_2(x_1)\}$$

x_1	$f_1(s, x_1) = c_{s,x_1} + f_2(x_1)$			Solusi Optimum	
$s \backslash$	2	3	4	$f_1(s)$	x_1^*
1	13	11	11	11	3 atau 4

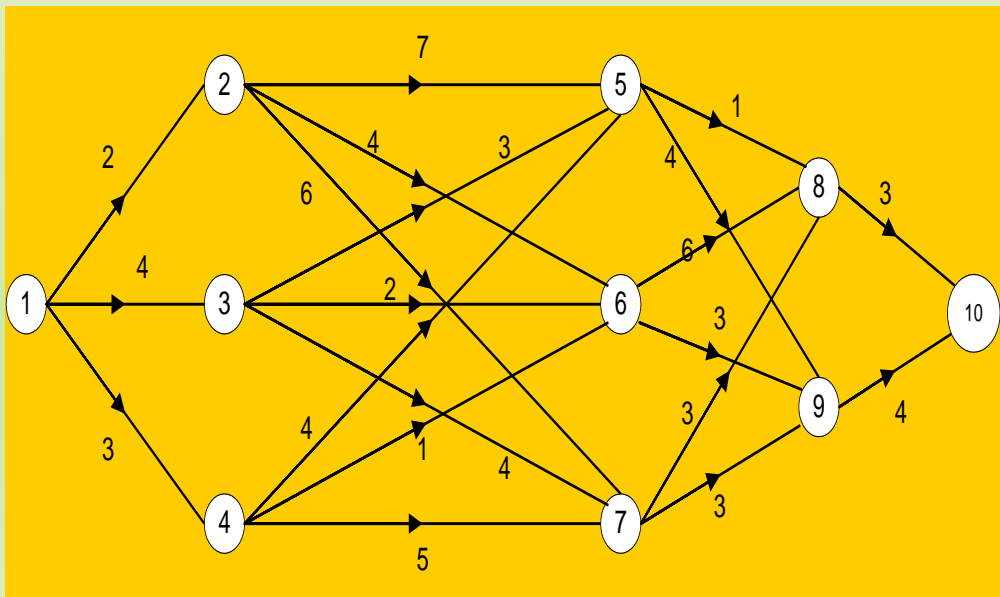


1. Lintasan Terpendek dengan PD Mundur

Tahap 1:

$$f_1(s) = \min_{x_1} \{c_{sx_1} + f_2(x_1)\}$$

x_1	$f_1(s, x_1) = c_{sx_1} + f_2(x_1)$			Solusi Optimum	
s	2	3	4	$f_1(s)$	x_1^*
1	13	11	11	11	3 atau 4



Solusi optimum dapat dibaca pada tabel di bawah ini:

	x_1	x_2	x_3	x_4	Panjang Lintasan Terpendek
1	3	5	8	10	11
	4	5	8	10	11
	4	6	9	10	11

Jadi ada tiga lintasan terpendek dari 1 ke 10, yaitu

$1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10$

$1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10$

$1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$

Panjang ketiga lintasan tersebut sama, yaitu 11.

2. Matrix-chain multiplication (MCM)

- Problem: given $\langle A_1, A_2, \dots, A_n \rangle$, compute the product: $A_1 \times A_2 \times \dots \times A_n$, find the fastest way (i.e., minimum number of multiplications) to compute it.
- Suppose two matrices $A(p,q)$ and $B(q,r)$, compute their product $C(p,r)$ in $p \times q \times r$ multiplications
 - for $i=1$ to p for $j=1$ to r $C[i,j]=0$
 - for $i=1$ to p
 - for $j=1$ to r
 - for $k=1$ to q $C[i,j] = C[i,j] + A[i,k]B[k,j]$

2. Matrix-chain multiplication (MCM)

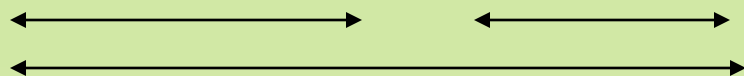
- Different parenthesizations will have different number of multiplications for product of multiple matrices
- Example: $A(10,100)$, $B(100,5)$, $C(5,50)$
 - If $((A \times B) \times C)$, $10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
 - If $(A \times (B \times C))$, $10 \times 100 \times 50 + 100 \times 5 \times 50 = 75000$
- The first way is ten times faster than the second !!!
- Denote $\langle A_1, A_2, \dots, A_n \rangle$ by $\langle p_0, p_1, p_2, \dots, p_n \rangle$
 - i.e, $A_1(p_0, p_1)$, $A_2(p_1, p_2)$, ..., $A_i(p_{i-1}, p_i)$, ..., $A_n(p_{n-1}, p_n)$

2. Matrix-chain multiplication (MCM)

- Intuitive brute-force solution: Counting the number of parenthesizations by exhaustively checking all possible parenthesizations.
- Let $P(n)$ denote the number of alternative parenthesizations of a sequence of n matrices:
 - $P(n) = \begin{cases} 1 & \text{if } n=1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$
- The solution to the recursion is $\Omega(2^n)$.
- So brute-force will not work.

2. Matrix-chain multiplication (MCM) - Steps

- **Step 1: structure of an optimal parenthesization**
 - Let $A_{i..j}$ ($i \leq j$) denote the matrix resulting from $A_i \times A_{i+1} \times \dots \times A_j$
 - Any parenthesization of $A_i \times A_{i+1} \times \dots \times A_j$ must split the product between A_k and A_{k+1} for some k , ($i \leq k < j$). The cost = # of computing $A_{i..k}$ + # of computing $A_{k+1..j}$ + # $A_{i..k} \times A_{k+1..j}$.
 - If k is the position for an optimal parenthesization, the parenthesization of “prefix” subchain $A_i \times A_{i+1} \times \dots \times A_k$ within this optimal parenthesization of $A_i \times A_{i+1} \times \dots \times A_j$ must be an optimal parenthesization of $A_i \times A_{i+1} \times \dots \times A_k$.
 - $A_i \times A_{i+1} \times \dots \times A_k \times A_{k+1} \times \dots \times A_j$



2. Matrix-chain multiplication (MCM)

- **Step 2: a recursive relation**

- Let $m[i,j]$ be the minimum number of multiplications for $A_i \times A_{i+1} \times \dots \times A_j$

$$m[i, j] = m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j .$$

- $m[1,n]$ will be the answer

$$m[i, j] = \begin{cases} 0 & \text{if } i = j , \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j\} & \text{if } i < j . \end{cases}$$

$$m[i, j] = \begin{cases} 0 & \text{if } i = j, \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j\} & \text{if } i < j. \end{cases}$$

2. Matrix-chain multiplication (MCM) - Steps

- **Step 3, Computing the optimal cost**
 - If by recursive algorithm, exponential time $\Omega(2^n)$ (ref. to P.346 for the proof.), no better than brute-force.
 - Total number of subproblems: $+n = \Theta(n^2)$
 - Recursive algorithm will encounter the same subproblem many times.
 - If tabling the answers for subproblems, each subproblem is only solved once.
 - The second hallmark of DP: **overlapping subproblems** and solve every subproblem just once.

2. Matrix-chain multiplication (MCM) - Steps

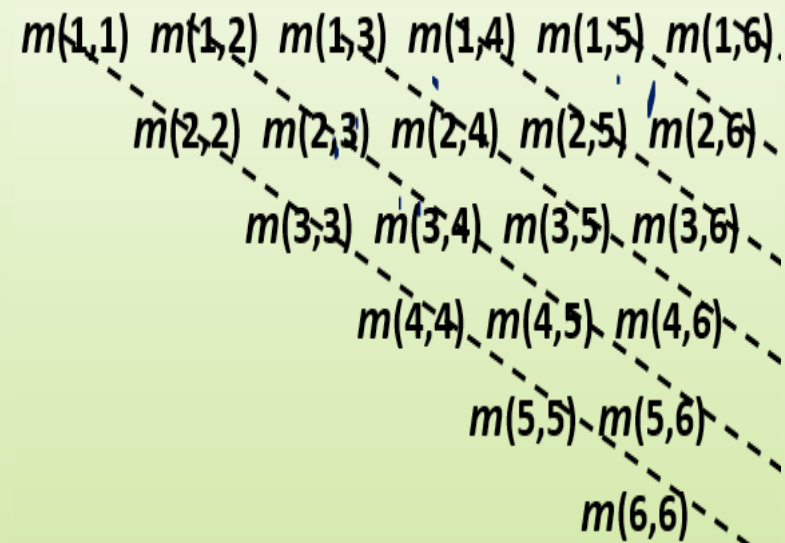
- **Step 3, Algorithm,**
 - array $m[1..n, 1..n]$, with $m[i, j]$ records the optimal cost for $A_i \times A_{i+1} \times \dots \times A_j$.
 - array $s[1..n, 1..n]$, $s[i, j]$ records index k which achieved the optimal cost when computing $m[i, j]$.
 - Suppose the input to the algorithm is $p = \langle p_0, p_1, \dots, p_n \rangle$.

2. MCM DP Steps

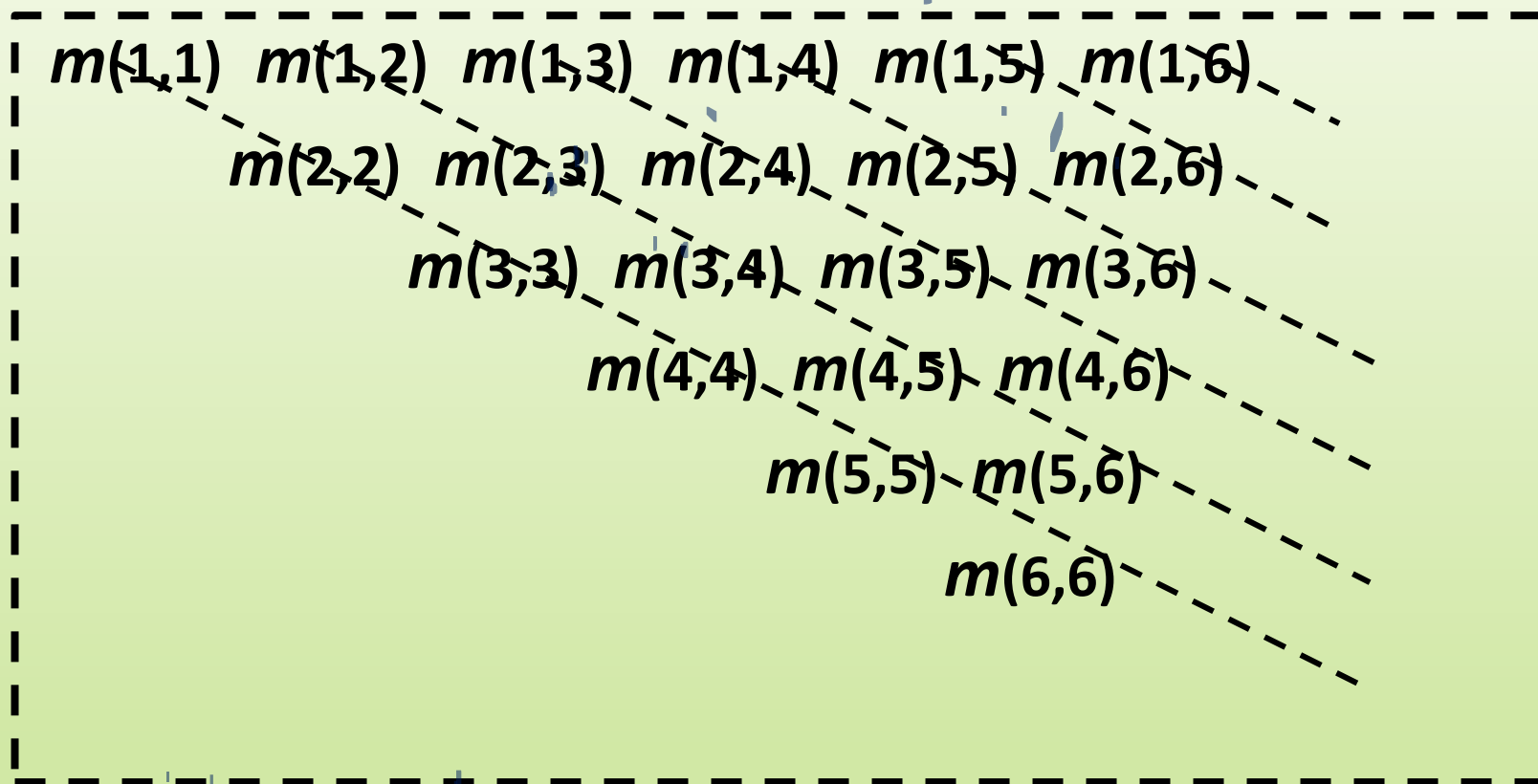
MATRIX-CHAIN-ORDER(p)

```

1   $n \leftarrow \text{length}[p] - 1$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      do  $m[i, i] \leftarrow 0$ 
4  for  $l \leftarrow 2$  to  $n$        $\triangleright l$  is the chain length.
5      do for  $i \leftarrow 1$  to  $n - l + 1$ 
6          do  $j \leftarrow i + l - 1$ 
7               $m[i, j] \leftarrow \infty$ 
8              for  $k \leftarrow i$  to  $j - 1$ 
9                  do  $q \leftarrow m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j$ 
10                     if  $q < m[i, j]$ 
11                         then  $m[i, j] \leftarrow q$ 
12                              $s[i, j] \leftarrow k$ 
13  return  $m$  and  $s$ 
```



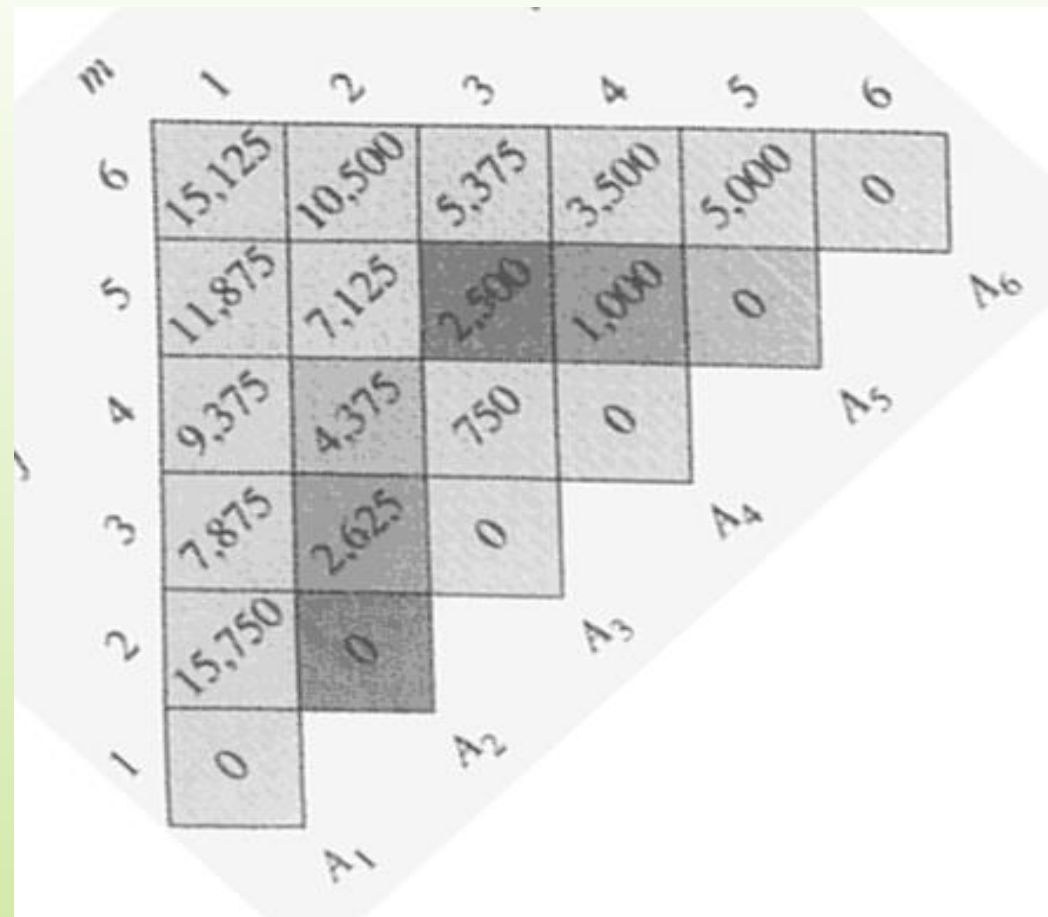
2. MCM DP—order of matrix computations



2. MCM DP Steps

Figure 15.3 The m and s tables computed by MATRIX-CHAIN-ORDER for $n = 6$ and the following matrix dimensions:

matrix	dimension
A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25



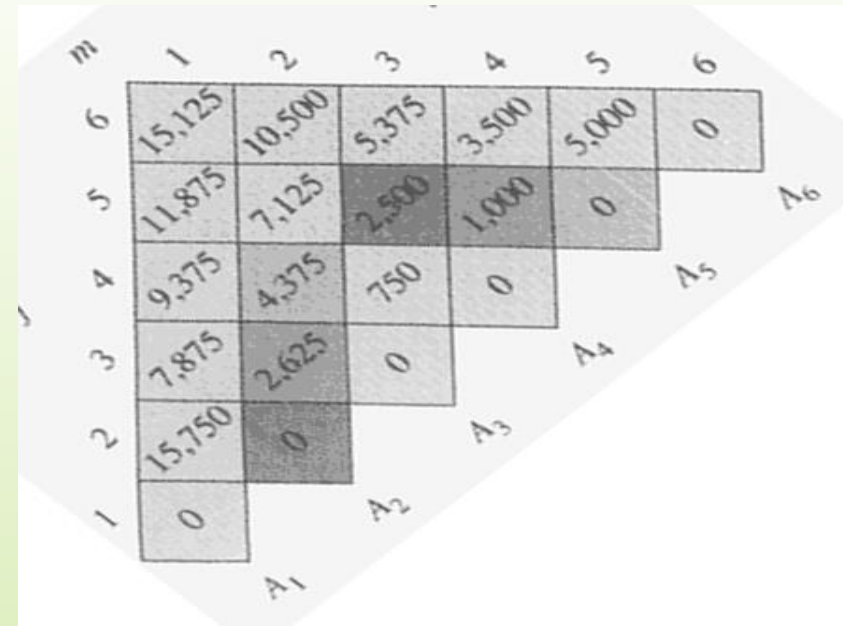
$$m[2, 5] = \min \begin{cases} m[2, 2] + m[3, 5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13000, \\ m[2, 3] + m[4, 5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125, \\ m[2, 4] + m[5, 5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11375 \end{cases} = 7125.$$

2. MCM DP Steps

MATRIX-CHAIN-ORDER(p)

```

1   $n \leftarrow \text{length}[p] - 1$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      do  $m[i, i] \leftarrow 0$ 
4  for  $l \leftarrow 2$  to  $n$        $\triangleright l$  is the chain length.
5      do for  $i \leftarrow 1$  to  $n - l + 1$ 
6          do  $j \leftarrow i + l - 1$ 
7               $m[i, j] \leftarrow \infty$ 
8              for  $k \leftarrow i$  to  $j - 1$ 
9                  do  $q \leftarrow m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j$ 
10                     if  $q < m[i, j]$ 
11                         then  $m[i, j] \leftarrow q$ 
12                              $s[i, j] \leftarrow k$ 
13  return  $m$  and  $s$ 
    
```



m	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					



s	1	2	3	4	5
6	3	3	3	5	5
5	3	3	3	4	
4	3	3	3		
3	1	2			
2	1				

2. MCM DP Steps

- Step 4, constructing a **parenthesization order** for the optimal solution.
 - Since $s[1..n, 1..n]$ is computed, and $s[i, j]$ is the split position for $A_i A_{i+1} \dots A_j$, i.e, $A_i \dots A_{s[i, j]}$ and $A_{s[i, j] + 1} \dots A_j$, thus, the **parenthesization order** can be obtained from $s[1..n, 1..n]$ recursively, beginning from $s[1, n]$.

PRINT-OPTIMAL-PARENS(s, i, j)

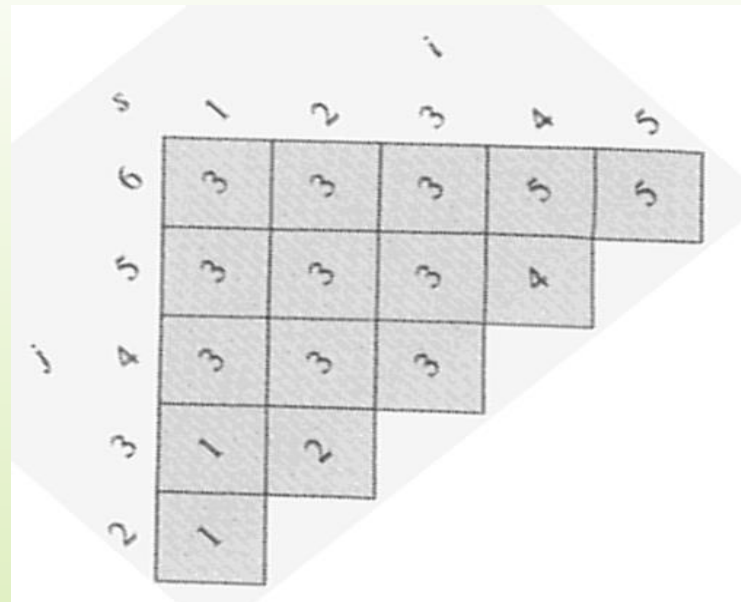
```
1  if  $i = j$ 
2      then print " $A$ ";
3  else print "("
4      PRINT-OPTIMAL-PARENS( $s, i, s[i, j]$ )
5      PRINT-OPTIMAL-PARENS( $s, s[i, j] + 1, j$ )
6      print ")"
```

PRINT-OPTIMAL-PARENS(s, i, j)

```

1  if  $i = j$ 
2      then print " $A$ ";
3  else print "("
4      PRINT-OPTIMAL-PARENS( $s, i, s[i, j]$ )
5      PRINT-OPTIMAL-PARENS( $s, s[i, j] + 1, j$ )
6  print ")"

```



Elements of DP

- **Optimal (sub)structure**
 - An optimal solution to the problem contains within it optimal solutions to subproblems.
- **Overlapping subproblems**
 - The space of subproblems is “small” in that a recursive algorithm for the problem solves the same subproblems over and over. Total number of distinct subproblems is typically polynomial in input size.
- **(Reconstruction an optimal solution)**

Characterize Subproblem Space

- Try to keep the space as simple as possible.
- In assembly-line schedule, $S_{1,j}$ and $S_{2,j}$ is good for subproblem space, no need for other more general space
- In matrix-chain multiplication, subproblem space $A_1A_2...A_j$ will not work. Instead, $A_iA_{i+1}...A_j$ (vary at both ends) works.

2. Recursive - Matrix-Chain Multiplication

RECURSIVE-MATRIX-CHAIN(p, i, j) (called with $(p, 1, n)$)

1. if $i=j$ then return 0
2. $m[i, j] \leftarrow \infty$
3. for $k \leftarrow i$ to $j-1$
4. do $q \leftarrow$ RECURSIVE-MATRIX-CHAIN(p, i, k) + RECURSIVE-MATRIX-CHAIN($p, k+1, j$) + $p_{i-1}p_kp_j$
5. if $q < m[i, j]$ then $m[i, j] \leftarrow q$
6. return $m[i, j]$

The running time of the algorithm is $O(2^n)$. Ref. to page 346 for proof.

Optimal Substructure Varies in Two Ways

- How many subproblems
 - n matrix-chain multiplication: two subproblems
- How many choices
 - In matrix-chain multiplication: $j-i$ choices
- DP solve the problem in bottom-up manner.
- #overall subproblems \times #choices.
 - In matrix-chain multiplication, $O(n^2) \times O(n) = O(n^3)$
- The cost = costs of solving subproblems + cost of making choice.
 - In matrix-chain multiplication, choice cost is $p_{i-1} p_k p_j$.

3. Longest Common Subsequence (LCS)

- DNA analysis, two DNA string comparison.
- DNA string: a sequence of symbols A,C,G,T.
 - $S = \text{ACCGGTCGAGCTTCGAAT}$
- Subsequence (of X): is X with some symbols left out.
 - $Z = \text{CGTC}$ is a subsequence of $X = \text{ACGCTAC}$.
- Common subsequence Z (of X and Y): a subsequence of X and also a subsequence of Y .
 - $Z = \text{CGA}$ is a common subsequence of both $X = \text{ACGCTAC}$ and $Y = \text{CTGACA}$.
- Longest Common Subsequence (LCS): the longest one of common subsequences.
 - $Z' = \text{CGCA}$ is the LCS of the above X and Y .
- LCS problem: given $X = \langle x_1, x_2, \dots, x_m \rangle$ and $Y = \langle y_1, y_2, \dots, y_n \rangle$, find their LCS.

3. LCS Intuitive Solution –brute force

- List all possible subsequences of X, check whether they are also subsequences of Y, keep the longer one each time.
- Each subsequence corresponds to a subset of the indices $\{1,2,\dots,m\}$, there are 2^m . So exponential.

3. LCS DP –step 1: Optimal Substructure

- Characterize optimal substructure of LCS.
- Theorem 15.1: Let $X = \langle x_1, x_2, \dots, x_m \rangle (= X_m)$ and $Y = \langle y_1, y_2, \dots, y_n \rangle (= Y_n)$ and $Z = \langle z_1, z_2, \dots, z_k \rangle (= Z_k)$ be any LCS of X and Y ,
 - 1. if $x_m = y_n$, then $z_k = x_m = y_n$, and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1} .
 - 2. if $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is the LCS of X_{m-1} and Y_n .
 - 3. if $x_m \neq y_n$, then $z_k \neq y_n$ implies Z is the LCS of X_m and Y_{n-1} .

3. LCS DP –step 2:Recursive Solution

- What the theorem says:
 - If $x_m = y_n$, find LCS of X_{m-1} and Y_{n-1} , then append x_m .
 - If $x_m \neq y_n$, find LCS of X_{m-1} and Y_n and LCS of X_m and Y_{n-1} , take which one is longer.
- Overlapping substructure:
 - Both LCS of X_{m-1} and Y_n and LCS of X_m and Y_{n-1} will need to solve LCS of X_{m-1} and Y_{n-1} .
- $c[i,j]$ is the length of LCS of X_i and Y_j .
$$c[i,j] = \begin{cases} 0 & \text{if } i=0, \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j, \\ \max\{c[i-1,j], c[i,j-1]\} & \text{if } i,j>0 \text{ and } x_i \neq y_j, \end{cases}$$

3. LCS DP-step 3:Computing the Length of LCS

- $c[0..m,0..n]$, where $c[i,j]$ is defined as above.
 - $c[m,n]$ is the answer (length of LCS).
- $b[1..m,1..n]$, where $b[i,j]$ points to the table entry corresponding to the optimal subproblem solution chosen when computing $c[i,j]$.
 - From $b[m,n]$ backward to find the LCS.

		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
i	x_i								
0			0	0	0	0	0	0	0
1	A		0	↑	↑	↑	↖1	←1	↖1
2	B		0	↖1	↑	←1	↑	↖2	←2
3	C		0	↑	↑	↖2	←2	↑	↑
4	B		0	↖1	↑	↑	↑	↖3	←3
5	D		0	↑	↖2	↑	↑	↑	↑
6	A		0	↑	↑	↑	↖3	↑	↖4
7	B		0	↖1	↑	↑	↑	↖4	↑

Figure 15.6 The c and b tables computed by LCS-LENGTH on the sequences $X = \langle A, B, C, B, D, A, B \rangle$ and $Y = \langle B, D, C, A, B, A \rangle$. The square in row i and column j contains the value of $c[i, j]$ and the appropriate arrow for the value of $b[i, j]$. The entry 4 in $c[7, 6]$ —the lower right-hand corner of the table—is the length of an LCS $\langle B, C, B, A \rangle$ of X and Y . For $i, j > 0$, entry $c[i, j]$ depends only on whether $x_i = y_j$ and the values in entries $c[i - 1, j]$, $c[i, j - 1]$, and $c[i - 1, j - 1]$, which are computed before $c[i, j]$. To reconstruct the elements of an LCS, follow the $b[i, j]$ arrows from the lower right-hand corner; the path is shaded. Each “↖” on the path corresponds to an entry (highlighted) for which $x_i = y_j$ is a member of an LCS.

3. LCS DP Algorithm

```
LCS-LENGTH( $X, Y$ )
1   $m \leftarrow \text{length}[X]$ 
2   $n \leftarrow \text{length}[Y]$ 
3  for  $i \leftarrow 1$  to  $m$ 
4      do  $c[i, 0] \leftarrow 0$ 
5  for  $j \leftarrow 0$  to  $n$ 
6      do  $c[0, j] \leftarrow 0$ 
7  for  $i \leftarrow 1$  to  $m$ 
8      do for  $j \leftarrow 1$  to  $n$ 
9          do if  $x_i = y_j$ 
10             then  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
11                  $b[i, j] \leftarrow \nwarrow$ 
12             else if  $c[i - 1, j] \geq c[i, j - 1]$ 
13                 then  $c[i, j] \leftarrow c[i - 1, j]$ 
14                      $b[i, j] \leftarrow \uparrow$ 
15                 else  $c[i, j] \leftarrow c[i, j - 1]$ 
16                      $b[i, j] \leftarrow \leftarrow$ 
17  return  $c$  and  $b$ 
```

3. LCS DP –step 4: Constructing LCS

```
PRINT-LCS( $b, X, i, j$ )  
1  if  $i = 0$  or  $j = 0$   
2      then return  
3  if  $b[i, j] = \nwarrow$   
4      then PRINT-LCS( $b, X, i - 1, j - 1$ )  
5          print  $x_i$   
6  elseif  $b[i, j] = \uparrow$   
7      then PRINT-LCS( $b, X, i - 1, j$ )  
8  else PRINT-LCS( $b, X, i, j - 1$ )
```

3. LCS space saving version

- Remove array b.
- **Print_LCS_without_b(c,X,i,j){**
 - If (i=0 or j=0) return;
 - If (c[i,j]==c[i-1,j-1]+1)
 - {Print_LCS_without_b(c,X,i-1,j-1); print x_i}
 - else if(c[i,j]==c[i-1,j])
 - {Print_LCS_without_b(c,X,i-1,j);}
 - else
 - {Print_LCS_without_b(c,X,i,j-1);}
- **}**
- **Can We do better?**
 - 2*min{m,n} space, or even min{m,n}+1 space for just LCS value.

4. Penganggaran Modal (*Capital Budgeting*)

- ❖ Sebuah perusahaan berencana akan mengembangkan usaha (proyek) melalui ketiga buah pabrik (*plant*) yang dimilikinya. Setiap pabrik diminta mengirimkan proposal (boleh lebih dari satu) ke perusahaan untuk proyek yang akan dikembangkan. Setiap proposal memuat total biaya yang dibutuhkan (c) dan total keuntungan (*revenue*) yang akan diperoleh (R) dari pengembangan usaha itu. Perusahaan mengalokasikan Rp 5 milyar untuk alokasi dana bagi ketiga pabriknya itu.

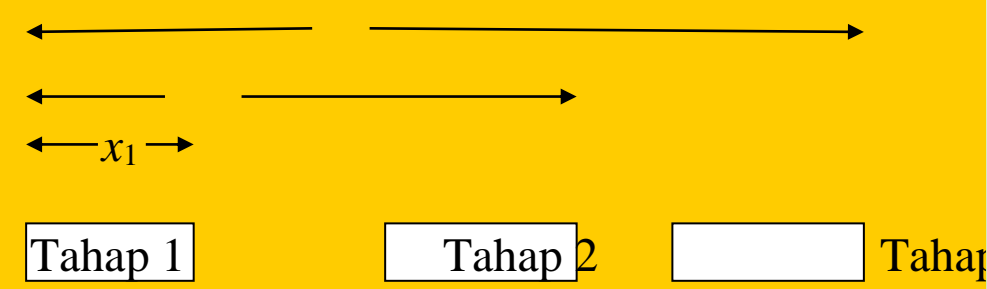
4. Penganggaran Modal (*Capital Budgeting*)

- ❖ Tabel berikut meringkaskan nilai c dan R untuk masing-masing proposal proyek. Proposal proyek bernilai-nol sengaja dicantumkan yang berarti tidak ada alokasi dana yang diberikan untuk setiap pabrik. Tujuan Perusahaan adalah memperoleh keuntungan yang maksimum dari pengalokasian dana sebesar Rp 5 milyar tersebut. Selesaikan persoalan ini dengan program dinamis.

4. Penganggaran Modal (*Capital Budgeting*)

Peubah status yang terdapat pada tahap 1, 2, dan 3:

$x_1 = \sum$ modal yang dialokasikan pada tahap 1
 $x_2 = \sum$ modal yang dialokasikan pada tahap 1 dan 2
 $x_3 = \sum$ modal yang dialokasikan pada tahap 1, 2, dan 3



Tahap 1 Tahap 2 Tahap 3

Kemungkinan nilai-nilai untuk x_1 dan x_2 adalah 0, 1, 2, 3, 4, 5 (milyar), sedangkan nilai untuk x_3 adalah 5

Proyek	Pabrik 1		Pabrik 2		Pabrik 3	
	c_1	R_1	c_2	R_2	c_3	R_3
1	0	0	0	0	0	0
2	1	5	2	8	1	3
3	2	6	3	9	-	-
4	-	-	4	12	-	-

4. Penganggaran Modal – Solusi PD Maju

Misalkan,

$R_k(p_k)$ = keuntungan dari alternatif p_k pada tahap k

$f_k(x_k)$ = keuntungan optimal dari tahap 1, 2, ..., dan k yang diberikan oleh status x_k

4. Penganggaran Modal - Solusi dengan PD

- ❖ Tahap (k) adalah proses mengalokasikan dana untuk setiap pabrik (ada 3 tahap, tiap pabrik mendefinisikan sebuah tahap).
- ❖ Status (xk) menyatakan jumlah modal yang dialokasikan pada pada setiap tahap (namun terikat bersama semua tahap lainnya).
- ❖ Alternatif (p) menyatakan proposal proyek yang diusulkan setiap pabrik. Pabrik 1, 2, dan 3 masing-masing memiliki 3, 4 dan 2 alternatif proposal.

4. Penganggaran Modal - Solusi dengan PD

Relasi rekurens keuntungan optimal:

$$f_1(x_1) = \max_{\substack{\text{feasible} \\ \text{proposal } p_1}} \{R_1(p_1)\} \quad (\text{basis})$$

$$f_k(x_k) = \max_{\substack{\text{feasible} \\ \text{proposal } p_k}} \{R_k(p_k) + f_{k-1}(x_{k-1})\} \quad (\text{rekurens})$$

$k = 2, 3$

Catatan:

1. $x_{k-1} = x_k - c_k(p_k)$
 $c(p_k)$ adalah biaya untuk alternatif p_k pada tahap k .
2. Proposal p_k dikatakan layak (*feasible*) jika biayanya, $c(p_k)$, tidak melebihi nilai status x_k pada tahap k .

Relasi rekurens keuntungan optimal menjadi

$$f_1(x_1) = \max_{c_1(p_1) \leq x_1} \{R_1(p_1)\} \quad (\text{basis})$$

$$f_k(x_k) = \max_{c_k(p_k) \leq x_k} \{R_k(p_k) + f_{k-1}[x_k - c_k(p_k)]\} \quad (\text{rekurens})$$

$k = 2, 3$

x_1	$R_1(p_1)$			Solusi Optimal	
	$p_1 = 1$	$p_1 = 2$	$p_1 = 3$	$f_1(x_1)$	p_1^*
0	0	-	-	0	1
1	0	5	-	5	2
2	0	5	6	6	3
3	0	5	6	6	3
4	0	5	6	6	3
5	0	5	6	6	3

4. Penganggaran Modal - Solusi dengan PD

Tahap 1

$$f_1(x_1) = \max_{\substack{c_1(p_1) \leq x_1 \\ p_1=1,2,3}} \{R_1(p_1)\}$$

x_1	$R_1(p_1)$			Solusi Optimal	
	$p_1 = 1$	$p_1 = 2$	$p_1 = 3$	$f_1(x_1)$	p_1^*
0	0	-	-	0	1
1	0	5	-	5	2
2	0	5	6	6	3
3	0	5	6	6	3
4	0	5	6	6	3
5	0	5	6	6	3

<u>Proyek</u>	<u>Pabrik 1</u>		<u>Pabrik 2</u>		<u>Pabrik 3</u>	
	c_1	R_1	c_2	R_2	c_3	R_3
1	0	0	0	0	0	0
2	1	5	2	8	1	3
3	2	6	3	9	-	-
4	-	-	4	12	-	-

4. Penganggaran Modal - Solusi dengan PD

<u>Proyek</u>	<u>Pabrik 1</u>		<u>Pabrik 2</u>		<u>Pabrik 3</u>	
	c_1	R_1	c_2	R_2	c_3	R_3
1	0	0	0	0	0	0
2	1	5	2	8	1	3
3	2	6	3	9	-	-
4	-	-	4	12	-	-

Tahap 1

$$f_1(x_1) = \max_{\substack{c_1(p_1) \leq x_1 \\ p_1=1,2,3}} \{R_1(p_1)\}$$

x_1	$R_1(p_1)$			Solusi Optimal	
	$p_1 = 1$	$p_1 = 2$	$p_1 = 3$	$f_1(x_1)$	p_1^*
0	0	-	-	0	1
1	0	5	-	5	2
2	0	5	6	6	3
3	0	5	6	6	3
4	0	5	6	6	3
5	0	5	6	6	3

Tahap 2

$$f_2(x_2) = \max_{\substack{c_2(p_2) \leq x_2 \\ p_2=1,2,3,4}} \{R_2(p_2) + f_1[(x_2 - c_2(p_2))]\}$$

x_2	$R_2(p_2) + f_1[(x_2 - c_2(p_2))]$ $R_2(p_2) + f_1[x_1]$				Solusi Optimal	
	$p_2 = 1$	$p_2 = 2$	$p_2 = 3$	$p_2 = 4$	$f_2(x_2)$	p_2^*
0	$0 + 0 = \mathbf{0}$	-	-	-	0	1
1	$0 + 5 = \mathbf{5}$	-	-	-	5	1
2	$0 + 6 = 6$	$8 + 0 = \mathbf{8}$	-	-	8	2
3	$0 + 6 = 6$	$8 + 5 = \mathbf{13}$	$9 + 0 = 9$	-	13	2
4	$0 + 6 = 6$	$8 + 6 = \mathbf{14}$	$9 + 5 = \mathbf{14}$	$12 + 0 = 12$	14	2 atau 3
5	$0 + 6 = 6$	$8 + 6 = 14$	$9 + 6 = 15$	$12 + 5 = \mathbf{17}$	17	4

4. Penganggaran Modal - Solusi dengan PD

Tahap 2

$$f_2(x_2) = \max_{\substack{c_2(p_2) \leq x_2 \\ p_2=1,2,3,4}} \{R_2(p_2) + f_1[(x_2 - c_2(p_2))]\},$$

x_2	$R_2(p_2) + f_1[(x_2 - c_2(p_2))]$ $R_2(p_2) + f_1[x_1]$				Solusi Optimal	
	$p_2 = 1$	$p_2 = 2$	$p_2 = 3$	$p_2 = 4$	$f_2(x_2)$	p_2^*
0	$0 + 0 = 0$	-	-	-	0	1
1	$0 + 5 = 5$	-	-	-	5	1
2	$0 + 6 = 6$	$8 + 0 = 8$	-	-	8	2
3	$0 + 6 = 6$	$8 + 5 = 13$	$9 + 0 = 9$	-	13	2
4	$0 + 6 = 6$	$8 + 6 = 14$	$9 + 5 = 14$	$12 + 0 = 12$	14	2 atau 3
5	$0 + 6 = 6$	$8 + 6 = 14$	$9 + 6 = 15$	$12 + 5 = 17$	17	4

<u>Proyek</u>	<u>Pabrik 1</u>		<u>Pabrik 2</u>		<u>Pabrik 3</u>	
	c_1	R_1	c_2	R_2	c_3	R_3
1	0	0	0	0	0	0
2	1	5	2	8	1	3
3	2	6	3	9	-	-
4	-	-	4	12	-	-

Tahap 3

$$f_3(x_3) = \max_{\substack{c_3(p_3) \leq x_3 \\ p_3=1,2}} \{R_3(p_3) + f_2[(x_3 - c_3(p_3))]\},$$

x_3	$R_3(p_3) + f_2[(x_3 - c_3(p_3))]$		Solusi Optimal	
	$p_3 = 1$	$p_3 = 2$	$f_3(x_3)$	p_3^*
5	$0 + 17 = 17$	$3 + 14 = 17$	17	1 atau 2

4. Penganggaran Modal - Solusi dengan PD

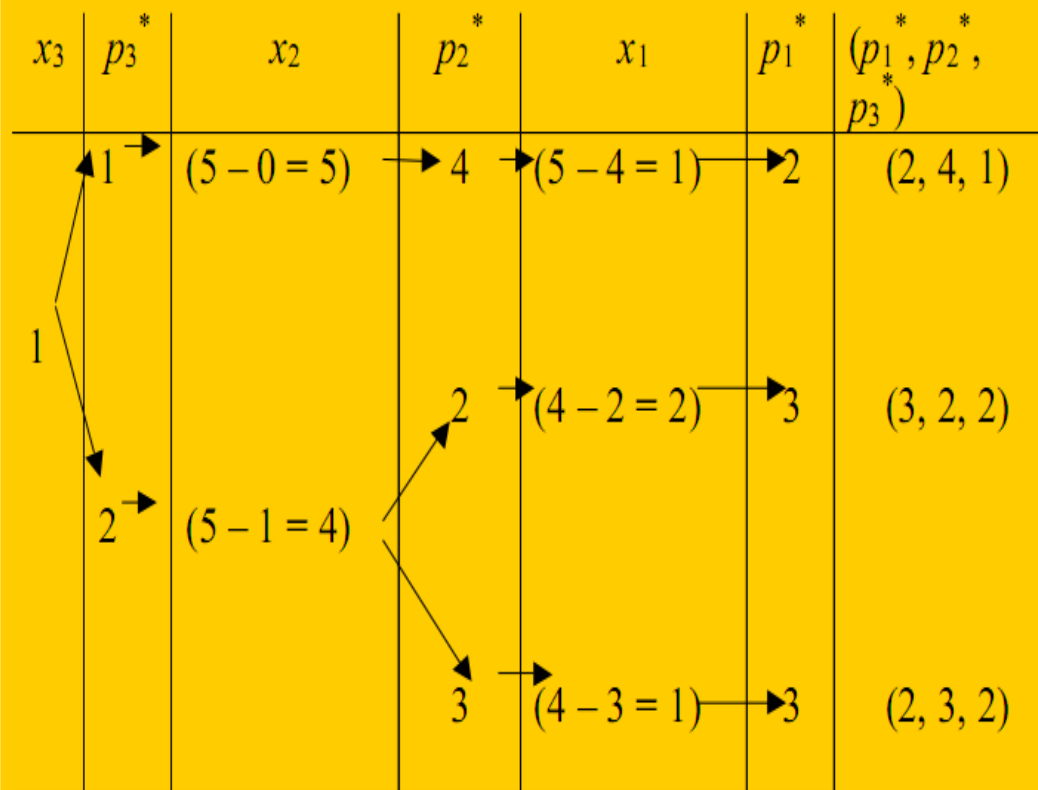
Proyek	Pabrik 1		Pabrik 2		Pabrik 3	
	c_1	R_1	c_2	R_2	c_3	R_3
1	0	0	0	0	0	0
2	1	5	2	8	1	3
3	2	6	3	9	-	-
4	-	-	4	12	-	-

Tahap 3

$$f_3(x_3) = \max_{\substack{c_3(p_3) \leq x_3 \\ p_3=1,2}} \{R_3(p_3) + f_2[(x_3 - c_3(p_3))]\},$$

x_3	$R_3(p_3) + f_2[(x_3 - c_3(p_3))]$		Solusi Optimal	
	$p_3 = 1$	$p_3 = 2$	$f_3(x_3)$	p_3^*
5	$0 + 17 = 17$	$3 + 14 = 17$	17	1 atau 2

Rekonstruksi solusi:



5. TSP (*Travelling Salesman Problem*)

- ❖ Misalkan $G = (V, E)$ adalah graf lengkap berarah dengan sisi-sisi yang diberi harga $c_{ij} > 0$.
- ❖ Misalkan $|V| = n$ dan $n > 1$. Setiap simpul diberi nomor 1, 2, ..., n .
- ❖ Asumsikan perjalanan (tur) dimulai dan berakhir pada simpul 1.
- ❖ Setiap tur pasti terdiri dari sisi $(1, k)$ untuk beberapa $k \in V - \{1\}$ dan sebuah lintasan dari simpul k ke simpul 1.
- ❖ Lintasan dari simpul k ke simpul 1 tersebut melalui setiap simpul di dalam $V - \{1, k\}$ tepat hanya sekali.

5. TSP (*Travelling Salesman Problem*)

❖ Prinsip Optimalitas

- ❖ Jika tur tersebut optimal maka lintasan dari simpul k ke simpul 1 juga menjadi lintasan k ke 1 terpendek yang melalui simpul-simpul di dalam $V - \{1, k\}$.
- ❖ Misalkan $f(i, S)$ adalah bobot lintasan terpendek yang berawal pada simpul i , yang melalui semua simpul di dalam S dan berakhir pada simpul 1. Maka Nilai $f(1, V - \{1\})$ adalah bobot tur terpendek.

5. TSP-PD

Hubungan rekursif:

$$f(1, V - \{1\}) = \min_{2 \leq k \leq n} \{c_{1k} + f(k, V - \{1, k\})\} \quad (1)$$

Dengan merampatkan persamaan (1), diperoleh

$$f(i, \emptyset) = c_{i,1} \quad , \quad 2 \leq i \leq n \quad \text{(basis)}$$

$$f(i, S) = \min_{j \in S} \{c_{ij} + f(j, S - \{j\})\} \quad \text{(rekurens)} \quad (2)$$

Gunakan persamaan (2) untuk memperoleh $f(i, S)$ untuk $|S| = 1$, $f(i, S)$ untuk $|S| = 2$, dan seterusnya sampai untuk $|S| = n - 1$.

5. TSP-PD

Soal : Misalkan diberikan persolaan TSM untuk $n=4$

0	10	15	20
5	0	9	10
6	13	0	12
8	8	9	0

Jawab :

- Dengan dengan algoritma PD

Tahap 1: $f(i, \emptyset) = c_{i1}$, $2 \leq i \leq n$

$$\begin{aligned} f(2, \emptyset) &= c_{21} = 5; \\ f(3, \emptyset) &= c_{31} = 6; \\ f(4, \emptyset) &= c_{41} = 8; \end{aligned}$$

Tahap 2:

$$f(i, S) = \min_{j \in S} \{c_{ij} + f(j, S - \{j\})\} \quad \text{untuk } |S| = 1$$

Diperoleh:

$$\begin{aligned} f(2, \{3\}) &= \min \{c_{23} + f(3, \emptyset)\} = \min \{9 + 6\} = \min \{15\} = 15 \\ f(3, \{2\}) &= \min \{c_{32} + f(2, \emptyset)\} = \min \{13 + 5\} = \min \{18\} = 18 \\ f(4, \{2\}) &= \min \{c_{42} + f(2, \emptyset)\} = \min \{8 + 5\} = \min \{13\} = 13 \\ f(2, \{4\}) &= \min \{c_{24} + f(4, \emptyset)\} = \min \{10 + 8\} = \min \{18\} = 18 \\ f(3, \{4\}) &= \min \{c_{34} + f(4, \emptyset)\} = \min \{12 + 8\} = \min \{20\} = 20 \\ f(4, \{3\}) &= \min \{c_{43} + f(3, \emptyset)\} = \min \{9 + 6\} = \min \{15\} = 15 \end{aligned}$$

5. TSP-PD

Tahap 2:

$$f(i, S) = \min_{j \in S} \{c_{ij} + f(j, S - \{j\})\} \quad \text{untuk } |S| = 1$$

Diperoleh:

$$f(2, \{3\}) = \min \{c_{23} + f(3, \emptyset)\} = \min \{9 + 6\} = \min \{15\} = 15$$

$$f(3, \{2\}) = \min \{c_{32} + f(2, \emptyset)\} = \min \{13 + 5\} = \min \{18\} = 18$$

$$f(4, \{2\}) = \min \{c_{42} + f(2, \emptyset)\} = \min \{8 + 5\} = \min \{13\} = 13$$

$$f(2, \{4\}) = \min \{c_{24} + f(4, \emptyset)\} = \min \{10 + 8\} = \min \{18\} = 18$$

$$f(3, \{4\}) = \min \{c_{34} + f(4, \emptyset)\} = \min \{12 + 8\} = \min \{20\} = 20$$

$$f(4, \{3\}) = \min \{c_{43} + f(3, \emptyset)\} = \min \{9 + 6\} = \min \{15\} = 15$$

0	10	15	20
5	0	9	10
6	13	0	12
8	8	9	0

Tahap 3:

$$f(i, S) = \min_{j \in S} \{c_{ij} + f(j, S - \{j\})\}$$

untuk $|S| = 2$ dan $i \neq 1, 1 \notin S$ dan $i \notin S$.

Diperoleh:

$$\begin{aligned} f(2, \{3, 4\}) &= \min \{c_{23} + f(3, \{4\}), c_{24} + f(4, \{3\})\} \\ &= \min \{9 + 20, 10 + 15\} \\ &= \min \{29, 25\} = 25 \end{aligned}$$

$$\begin{aligned} f(3, \{2, 4\}) &= \min \{c_{32} + f(2, \{4\}), c_{34} + f(4, \{2\})\} \\ &= \min \{13 + 18, 12 + 13\} \\ &= \min \{31, 25\} = 25 \end{aligned}$$

$$\begin{aligned} f(4, \{2, 3\}) &= \min \{c_{42} + f(2, \{3\}), c_{43} + f(3, \{2\})\} \\ &= \min \{8 + 15, 9 + 18\} \\ &= \min \{23, 27\} = 23 \end{aligned}$$

5. TSP-PD

Tahap 3:

$$f(i, S) = \min_{j \in S} \{c_{ij} + f(j, S - \{j\})\}$$

untuk $|S| = 2$ dan $i \neq 1, 1 \notin S$ dan $i \notin S$.

Diperoleh:

$$\begin{aligned} f(2, \{3, 4\}) &= \min \{c_{23} + f(3, \{4\}), c_{24} + f(4, \{3\})\} \\ &= \min \{9 + 20, 10 + 15\} \\ &= \min \{29, 25\} = 25 \end{aligned}$$

$$\begin{aligned} f(3, \{2, 4\}) &= \min \{c_{32} + f(2, \{4\}), c_{34} + f(4, \{2\})\} \\ &= \min \{13 + 18, 12 + 13\} \\ &= \min \{31, 25\} = 25 \end{aligned}$$

$$\begin{aligned} f(4, \{2, 3\}) &= \min \{c_{42} + f(2, \{3\}), c_{43} + f(3, \{2\})\} \\ &= \min \{8 + 15, 9 + 18\} \\ &= \min \{23, 27\} = 23 \end{aligned}$$

Hubungan rekursif:

$$f(1, V - \{1\}) = \min_{2 \leq k \leq n} \{c_{1k} + f(k, V - \{1, k\})\} \quad (1)$$

Dengan menggunakan persamaan (1) diperoleh:

$$\begin{aligned} f(1, \{2, 3, 4\}) &= \min \{c_{12} + f(2, \{3, 4\}), c_{13} + f(3, \{2, 4\}), \\ &\quad c_{14} + f(4, \{2, 3\})\} \\ &= \min \{10 + 25, 15 + 25, 20 + 23\} \\ &= \min \{35, 40, 43\} = 35 \end{aligned}$$

Jadi, bobot tur yang berawal dan berakhir di simpul 1 adalah 35.

5. TSP-PD

- ❖ Misalkan $J(i, S)$ adalah nilai yang dimaksudkan tersebut. Maka, $J(1, \{2, 3, 4\}) = 2$. Jadi, tur mulai dari simpul 1 selanjutnya ke simpul 2.
- ❖ Simpul berikutnya dapat diperoleh dari $f(2, \{3, 4\})$, yang mana $J(2, \{3, 4\}) = 4$. Jadi, simpul berikutnya adalah simpul 4.
- ❖ Simpul terakhir dapat diperoleh dari $f(4, \{3\})$, yang mana $J(4, \{3\}) = 3$. Jadi, tur yang optimal adalah 1, 2, 4, 3, 1 dengan bobot (panjang) = 35.

Hubungan rekursif:

$$f(1, V - \{1\}) = \min_{2 \leq k \leq n} \{c_{1k} + f(k, V - \{1, k\})\} \quad (1)$$

Dengan menggunakan persamaan (1) diperoleh:

$$\begin{aligned} f(1, \{2, 3, 4\}) &= \min \{c_{12} + f(2, \{3, 4\}), c_{13} + f(3, \{2, 4\}), \\ &\quad c_{14} + f(4, \{2, 3\})\} \\ &= \min \{10 + 25, 15 + 25, 20 + 23\} \\ &= \min \{35, 40, 43\} = 35 \end{aligned}$$

Jadi, bobot tur yang berawal dan berakhir di simpul 1 adalah 35.

0	10	15	20
5	0	9	10
6	13	0	12
8	8	9	0

- Click to edit subtitle style

Thank You !